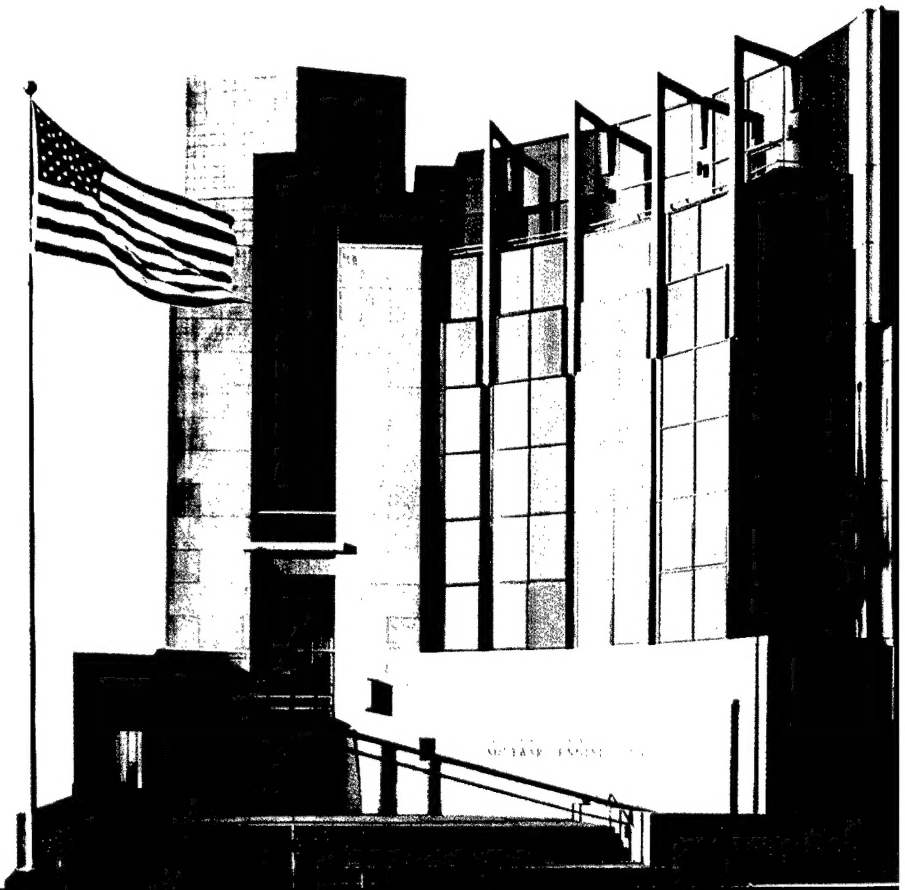
Carnegie Mellon
Software Engineering Institute

SEI Independent Research and Development Projects

Felix Bachmann
Len Bass
David Carney
Sven Dietrich
Peter Feiler
Suzanne Garcia
Mark Klein
Tony Lattanze
John McHugh
B. Craig Meyers
Ed Morris
Patrick R. H. Place
Dan Plakosh
Robert Seacord

September 2003



TECHNICAL REPORT
CMU/SEI-2003-TR-019
ESC-TR-2003-019

Carnegie Mellon
Software Engineering Institute

Pittsburgh, PA 15213-3890

SEI Independent Research and Development Projects

CMU/SEI-2003-TR-019
ESC-TR-2003-019

Felix Bachmann
Len Bass
David Carney
Sven Dietrich
Peter Feiler
Suzanne Garcia
Mark Klein
Tony Lattanze
John McHugh
B. Craig Meyers
Ed Morris
Patrick R. H. Place
Dan Plakosh
Robert Seacord

September 2003

SEI Director's Office

This report was prepared for the

SEI Joint Program Office
HQ ESC/DIB
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Christos Scondras
Chief of Programs, XPK

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

This work was created in the performance of Federal Government Contract Number F19628-00-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

Abstract vii

1 Introduction 1
1.1 Purpose of the SEI Independent Research and Development Program 1
1.1.1 Overview of IR&D Projects 2

2 Acquisition Simulation 3
2.1 Purpose 3
2.2 Background 3
2.3 Approach 4
2.4 Collaborators 5
2.5 Evaluation Criteria 6
2.6 Results 7
2.6.1 Present State of the Prototypes 8
2.6.2 Evaluation Against the Evaluation Criteria 9
2.6.3 Future Steps 9
2.7 Publications and Presentations 10

3 Architectural Design Assistant 11
3.1 Purpose 11
3.2 Background 11
3.3 Approach 13
3.4 Collaborators 14
3.5 Evaluation Criteria 14
3.6 Results 14
3.6.1 Clarifying the Theoretical Background 15
3.6.2 Preliminary Design of the Design Assistant 16
3.7 Publications and Presentations 17

4 A Model-Based Reference Architecture (MRA) for Mobile Robotics Systems 19
4.1 Purpose 19
4.2 Background 20

4.3 Approach 21
4.3.1 Phase I: Architecture Discovery 21
4.3.2 Phase II: Architecture Capture and Model-based Analysis 21
4.3.3 Phase III: Reference Architecture Validation 22
4.4 Collaborators 22
4.5 Evaluation Criteria 23
4.6 Results 24
4.6.1 Architecture Discovery 24
4.6.2 Architecture Capture and Model-Based Analysis 27
4.7 Publications and Presentations 27

5 Securing Wireless Devices 29
5.1 Purpose 29
5.2 Background 29
5.3 Collaborators 30
5.4 Results 30
5.5 Lessons Learned 31
5.6 Conclusion 32

6 Sustainment 35
6.1 Purpose 35
6.2 Background 36
6.2.1 Legacy System Crisis 36
6.2.2 Current SEI Efforts 36
6.3 Approach 37
6.3.1 Potential Impact 38
6.4 Collaborators 39
6.4.1 Research Collaborators 39
6.4.2 Advisory Board 39
6.5 Evaluation Criteria 39
6.6 Results 40
6.6.1 Sustainability Assessment 41
6.7 Publications 42

7 System of Systems Interoperability (SOSI) 43
7.1 Purpose 43
7.2 Background 44
7.3 Approach 44
7.3.1 Interoperability Model 45
7.3.2 Method 46
7.4 Collaborators 47

7.5 Evaluation Criteria 47
7.6 Results 48
7.6.1 Observations on the Model 48
7.6.2 Emergent Themes 48
7.6.3 Future State 50
7.7 Publications and Presentations 51

Bibliography 53

List of Figures

Figure 1: Module Decomposition View of Designer's Assistant 16

Figure 2: System Life-Cycle Costs 38

Figure 3: Sustainability Attribute 41

Figure 4: WMRD 41

Figure 5: System Activities Model 45

Figure 6: Different Types of Interoperability 46

Abstract

Each year, the Software Engineering Institute (SEI) undertakes several Independent Research and Development (IR&D) projects. These projects serve to (*a*) support feasibility studies investigating whether further work by the SEI would be of potential benefit and (*b*)support further exploratory work to determine if there is sufficient value in eventually funding the feasibility study work as an SEI initiative. Projects are chosen based on their potential to mature and/or transition software engineering practices, develop information that will help in deciding whether further work is worth funding, and set new directions for SEI work. This report describes the IR&D projects that were conducted during fiscal year 2003 (October 2002 through September 2003).

1 Introduction

1.1 Purpose of the SEI Independent Research and Development Program

Software Engineering Institute (SEI) Independent Research and Development (IR&D) funds are used in two ways: (1) to support feasibility studies investigating whether further work by the SEI would be of potential benefit and (2) to support further exploratory work to determine if there is sufficient value in eventually funding the feasibility study work as an SEI initiative. It is anticipated that each year there will be three or four feasibility studies and that one or two of these studies will be further funded to lay the basis for the work possibly becoming an initiative.

Feasibility studies are evaluated against the following criteria:

- Mission criticality: To what extent is there a potentially dramatic increase in maturing and/ or transitioning software engineering practices if work on the proposed topic yields positive results? What will the impact be on the DoD?
- Sufficiency of study results: To what extent will information developed by the study help in deciding whether further work is worth funding?
- New directions: To what extent does the work set new directions as contrasted with building on current work? (Ideally, we want a mix of studies that build on current work and studies that set new directions.)

At a DoD meeting in November 2001, the SEI's DoD sponsor approved a set of thrust areas and challenge problems to provide long-range guidance for the SEI R&D program, including its IR&D program. The thrust areas are survivability/security, interoperability, sustainability, software R&D, metrics for acquisition, acquisition management, and commercial off-the-shelf products. The IR&D projects conducted in FY2003 were based on these thrust areas and challenge problems.

1.1.1 Overview of IR&D Projects

The following research projects were undertaken in FY2003:

- Architectural Design Assistant
- Model-Based Reference Architectures for Mobile Robotics Systems
- Securing Wireless Devices
- Simulation-Based Acquisition Management Training
- Sustainment
- Systems of Systems Interoperability

These projects are described in detail in the same order as they are listed here.

2 Acquisition Simulation

2.1 Purpose

The problem that motivated this work was the lack of hands-on experience available during the training of acquisition personnel, a lack that has been evidenced in many of the SEI's Independent Technical Analyses (ITAs). At present, the only way that a government program manager can learn the reality and complexity of acquisition is by executing actual programs, during which decisions that affect millions of dollars are made, but with little or no practical experience on which to base them.

The purpose of this project was therefore to determine the feasibility of applying simulation, gaming, and role-playing techniques to the training of acquisition personnel. The potential impact of this work would be to better equip program managers with skills matched to the complexities of modern acquisition (e.g., negotiations with COTS vendors), to improve the realism of program cost and schedule forecasts, and to increase the awareness of program managers and other key persons about the dynamics that underlie typical acquisition problems.

What we sought to do was different from the more common use of simulation in government acquisition, i.e., simulation of a target system, such as flight simulators for an aircraft's pilot and crew. Rather than exploring simulation of a deliverable system, our interest was in simulating elements of the acquisition process itself, and our target audience was the government personnel that manage acquisitions.

2.2 Background

Simulation and role-play techniques are used as training mechanisms in many areas, e.g., public policy management, financial management, supply-chain management, and other comparable fields. Most educational institutions in these fields make extensive use of simulation and role-playing techniques in their curricula. For example, the CMU Graduate School of Industrial Administration, MIT's Sloan School, and numerous other institutions have for many years made simulation a principal teaching mechanism; students are required to participate in long-term simulations of common management problems, with the need to make complex decisions about real-life issues and events. One product of MIT, the "Beer Game" simulation, which deals with supply chain management, has been the subject of a very large

number of technical journal articles, seminars, and conference presentations. The Department of Defense, in its use of war games, is also a major proponent of using simulation to instill experiential knowledge of combat to military personnel in advance of actual warfare.

In addition to these efforts aimed at training, simulation techniques are being employed in a large number of other areas. For example, "Interactive Drama" is moving from the research lab to the commercial arena, and is expected to become significant in many commercial ventures (see http://www.interactivestory.net for information on one such effort, the "Façade" project). Research at University of Sweden has begun in "Game Patterns," similar to the Design Patterns work begun by Alexander that became very significant in the domain of computer architecture [Alexander 64]. A community focused on "serious games" is in the formative stages. And system dynamics modeling is actively used in both business simulations and commercial computer games (e.g., the popular computer game "SimCity").

A different genre of simulation was developed around 1993 by Richard Garfield, that of customizable card games (CCG). The first game of the genre, called Magic: The Gathering, still holds a strong position in the CCG market. A subset of this genre of particular interest to this study includes the "building" simulation games, such as the CCG version of SimCity. This mode of simulation adds portability and variety that can supplement other, more traditional forms of simulation.

Finally, and separate from games research per se, the domain of adult learning research has witnessed considerable work in simulation, since live role-play simulations generally take place in some kind of formal or informal educational or training context.

2.3 Approach

In choosing an approach, it was our expectation that since the SEI was instituting a broad-based initiative in acquisition, the result of our work should have some logical relation to the work of that initiative. A further support for this assumption was direction from Steve Cross, the former Director of SEI, that our work should contribute to "mission rehearsal workshops" for the acquisition community. Our approach was therefore pragmatic:

1. We would demonstrate that simulation was indeed applicable to the domain of acquisition through a number of prototypes that, although small in scale, would be sufficient to show how different modes of simulation could improve training of acquisition professionals.
2. We would not invent technology. Considerable simulation technology exists, both computer-based simulation as well as technology independent of any specific form of technical support. We chose to build on what exists and to leverage as much technology as possible.

3. We would not attempt to simulate "the acquisition process" as a whole,[1] but would take small slices of that process, and simulate small, bounded process fragments using various simulation technologies and modes. Our approach was thus narrow and deep, an approach which is different than, though complementary with, one of our collaborators (the Defense Acquisition University [DAU]), whose curriculum exhibits a wide and shallow approach.

4. Since there are many modes of simulation that are of potential value, we needed to choose among the several possible kinds of simulation we would use. For instance, simulations can be single player or multi-player; they can be computer-based or computer-supported or non-computer simulations (for example, CCGs). Other possible variables include asynchronous versus synchronous simulations, or single-winner versus multiple mutual goal simulations.

We decided that we should concentrate on a small number of relatively simple demonstrations that simulation techniques could be successfully used in the acquisition domain. The modes we chose were computer-supported simulations, both single-player and multi-player, and also multi-player CCGs. The choice of computer-based simulation was based on our primary interest in role-play as a key training mechanism for the acquisition domain. The decision to include multi-player CCGs provided a portable mode to continue and reinforce the learning acquired during the computer-supported simulations (most likely to occur in some sort of live workshop setting). There is anecdotal evidence that card games, in addition to offering portability, actually offer a welcome relief from the computer technology that has become a large part of the daily work of all acquisition office personnel. The card game also offered the potential for sharing learning with persons who were not present in the workshop setting.

2.4 Collaborators

The following people participated in this project:

- SEI MTS (principal investigators): David Carney, Suzanne Garcia
- Other SEI: (Alpha participants): Jack Ferguson, Brian Gallagher, Carolyn Graettinger, Jon Gross, Tom Miller, Mike Philips, Dennis Smith, Bob Vrtis
- Other CMU (student programmers): Lalit Jina, Shailu Mishra, Dehua Zhang, Goran Momiroski
- External collaborators (all providing their own support): Richard Adams (Aerospace), William Rouse (Georgia Tech), George Prosnik (Defense Acquisition University)

[1] It is difficult even to identify all of the elements of the large and complex process that is "the acquisition process."

The Alpha participants each devoted one full day of effort to the simulation run-through. The four CMU students did the programming on one of the prototypes. The external collaborators each had some significant overlap with our work. Richard Adams is investigating application of systems dynamics techniques to acquisition. William Rouse heads a large research effort examining use of simulation in various government and industry domains. And George Prosnik is working to improve acquisition education, including broader and deeper use of simulation as a teaching mechanism. For each external collaborator, we held one or more meetings to discuss avenues of mutual benefit. Of these, the relationship with the DAU holds the greatest promise, since we anticipate that if a full simulation facility were to be created, there would be some sort of joint stewardship between the SEI and the DAU in its operation.

2.5 Evaluation Criteria

We developed two kinds of evaluation criteria: those to evaluate the success of the IRAD, and those to evaluate the success of the prototypes. Although there is some overlap between them, we believed that this was a useful distinction.

If the IRAD were to successfully demonstrate that simulation and role-play could become a mainstream approach in the acquisition community, the following questions would apply:

- Domain applicability: Could the elements needed to create simulations of acquisition be specified adequately?
- Personal applicability: Could a significant subset of the people in the acquisition community be interested in or engaged by playing any sort of acquisition "game"? Said differently, to what extent is that community made up of "players" or "gamers"?
- Cost: what price point would be needed to recoup costs? Could we find a transition partner who would package or market the concepts that we investigated?

These questions relate to evaluating the success of the prototypes:

- Utility: Do the prototypes have applicability to real acquisition problems and situations? In particular, does the CCG effectively have such applicability?
- "Gamishness": Are the prototypes sufficiently enjoyable? Do they motivate players to continue to play the game?
- Efficiency: Does enacting the prototype impose huge demands on the participant or the agency that offers the prototype? Is it cost-effective in relation to the learning that is done?

2.6 Results

(This IRAD was still in progress as of the writing of this report, as it was not to be completed until the end of September.)

After an initial period of literature review, we concluded that we should focus on two bounded aspects of acquisition: the problem of requirements definition and issues related to contract award fee. For each of these, we created a limited scenario, defined a set of related artifacts, and created a set of roles, agents, constraints, and events. In defining the scenario for requirements definition we found it useful (at Steve Cross's suggestion) to base the scenario on an unpublished case study by Carter et al. that the SEI had prepared for use by the DAU.

We then considered the creation of three prototype products. We imagined that each of them could be implemented using either scenario. We chose the requirements scenario as the initial vehicle for the prototypes. Note that the award fee scenario is, we believe, an equally useful vehicle for the demonstration; because of time and resource constraints we were able to implement only the requirements scenario.

The three prototypes were

- a single-player "solitaire" computer game
- a multi-player role-play game, for multiple teams of participants, with extensive computer support
- a CCG version of the multi-player role-play game

The solitaire game, "The Requirements Game," has a loose kinship with the MIT "Beer Game" in that it can be repeatedly played by a single player. While repeated playings can permit the player to increase his skill with the game, new plays of the game will provide challenges even for an experienced player.

The multi-player role-play game was the major focus of our work. It uses a fairly complex scenario of a joint program with diverse stakeholders who have competing requirements; the point of the game is for participants to take on the roles of these different stakeholders and find solutions that will permit the desired system to be built.

The conception of the CCG was developed jointly with the multi-player role-play game, since our expectation was that the CCG would be closely related to the same scenario.

2.6.1 Present State of the Prototypes

The prototypes are in various stages of completion.

The solitaire "Requirements Game" is implemented in Macromedia Flash. The prototype is still in need of considerable programming in order to bring it to a state for alpha testing. It can be executed from start to finish, and complete games can be played. The user interface is in need of an overhaul, and the form in which games are created and notated needs to be automated and revised. However, even in its present form, it is a lightweight (though highly "buggy") demonstration of a simple simulation mechanism applied to genuine acquisition problems. It is still not possible to assess whether the game's goals and expectations have been met (i.e., the degree to which the game is both educational and "fun").

The development of the CCG progressed through preliminary design. At that point, however, we realized that it would be impossible to develop both it and the multi-player role-play simultaneously: to ensure the close relationships between two simulations, most of the content of the multi-player role-play needed to be available before implementation in the CCG. We chose therefore the concentrate on the role play and leave the card game for a future effort.

The multi-player role-play game was brought to a state sufficient to perform an alpha test. For that test we enlisted eight senior SEI personnel, all of whom had deep and experience-based understanding of DoD acquisition (see section 4.0 above). We conducted the alpha test over a full day, aided by considerable support from several persons in Information Technology at the SEI. We drew the following conclusions from the alpha test of the role-play:

- Participants were enthusiastic: all participants strongly urged us to perform another execution as quickly as possible. To that end, we are presently working with PID to identify at least one site to perform a beta test (see section 7.0).
- Artifacts of the simulation (i.e., those things that the scenario imposed on the participants to read, react to, etc.) were sufficiently accurate to give the participants a feeling that they were actually dealing with a "real" program.
- Execution of the scenario (i.e., the pace and speed with which the events occurred and the artifacts came into play) was too compact and too dense. The conclusion was that further experimentation would reveal a more optimal pacing and density for execution of the scenario.
- Activities that the participants could perform (i.e., the elements of the scenario that the participants could influence) were not sufficiently rich. The participants all agreed that the scenario forced them into too passive a posture. The conclusion was that further experimentation would reveal a more optimal balance between decisions by the participants and events that are pre-ordained by the scenario.

We are currently working on the beta version of the simulation, and expect to complete it by mid-September; more than one beta execution may prove necessary.

2.6.2 Evaluation Against the Evaluation Criteria

Given that we are still in process, not all of the evaluation criteria can be fully determined. However, the following appear to be true for evaluating the IRAD:

- Domain applicability: based on the feedback from the alpha test, the applicability of simulation to the acquisition domain has been amply demonstrated. (Note that the evidence from DAU also shows this very strongly.)
- Personal applicability: also based on feedback from the alpha test, acquisition personnel reacted very favorably to a game-based and role-play approach to acquisition issues.
- Cost: Computer-based simulations in the form of games are becoming more common, and their cost is diminishing. Nonetheless, they are still an expensive proposition; data from simulation conferences indicates that the cost of developing a computerized role-play game to the point of marketing it are roughly $10M. By contrast, our total costs were roughly $250K. We can only conjecture about a transition partner, but our intuition is that one could be found.

Since our only real evidence is based on the alpha test of the multi-player prototype, the three evaluations of the prototypes are only partial.

- Utility: The scenario used in the alpha test was very closely related to actual issues that arise in requirements definition. The participants especially praised the quality of the artifacts use in the simulation.
- "Gamishness": All of the participants reported that "playing the game" was an enjoyable exercise. The major problem in this regard was in the technical arena, since the complexity of the demands we placed on computer support were in excess of the available resources. We anticipate that this will be solved in the beta version.
- Efficiency: We anticipate that should this simulation become a product, its execution should be accomplished in half a day (as opposed to the full day for the alpha test). If this is true, then this should not be an undue expenditure of resources in relation to the expected educational value, since 8 to 12 people can participate in a single simulation event.

2.6.3 Future Steps

We hope to bring the multi-player role play to beta condition by the end of September, at which point this project will be completed. Presuming that one or both of the possible beta deliveries of the simulation goes forward, we then expect to propose to the Acquisition Support Program that further work be conducted to expand the simulation and role-playing prototype into an SEI product.

2.7 Publications and Presentations

We have not yet published any papers on this work. However, the planned publications include a Technical Report on the full scope of the project, and a technical note that focuses on the CCG aspect of the project. In addition, Suzanne Garcia delivered a presentation, "Acquisition Simulation Card Game: Ideas a Year Later," at the Consultant's Camp meeting in August, 2003. We are also preparing an annotated bibliography of simulation references and resources that will be useful for any continued effort in this area.

We are revising the scenario role-play simulation, based on the extensive feedback from the alpha participants, and are tentatively scheduled to give a beta presentation to a group of acquisition personnel at the Electronic Systems Center at Hansom AFB in September or October. We have also been in discussion with personnel from the Army's AMCOM program in Huntsville, and they have expressed interest in being a beta test site as well. If this proposal progresses, we hope to schedule that simulation for November or December.

3 Architectural Design Assistant

3.1 Purpose

The purpose of this work is to investigate the feasibility of using expert systems technology to codify, promulgate, and make accessible quality attribute knowledge to enhance software architecture design and analysis. The goal is to create a design assistant to enhance an ordinary software designer's capability so that he or she can perform at the level of an expert software designer with a specific emphasis on quality attribute-based design and analysis.

Creating an architectural design assistant would have the following benefits:

- codification of architecture design knowledge and hence promulgation of this knowledge as concrete guidance to designers and analysts
- the capture of this design knowledge into an easily modified knowledge base so that organizations would be able to augment the knowledge base with their own domain specifics
- a vehicle for "testing" the completeness of our work on codifying the relation between design decisions and quality attribute responses via tactics
- a "tireless quality attribute expert" that can simultaneously participate on multiple Architecture Tradeoff Analysis Method[SM] (ATAM[SM]) and Attribute Driven Design (ADD) exercises

3.2 Background

One of the principles of our work in architecture design and analysis is that the quality attribute requirements (such as performance, security, modifiability, reliability, and usability) exert a dominant influence on the "shape" of the architecture. In fact, we verify this principle every time we conduct an architecture evaluation using the ATAM or an architectural design using the ADD. With minimal domain knowledge, a team armed with relevant attribute knowledge can find architectural risks using the ATAM or make architectural design decisions to realize quality attribute goals.

[SM] Architecture Tradeoff Analysis Method and ATAM are service marks of Carnegie Mellon University.

An important observation about these methods is that, while they efficiently orchestrate the use of the quality knowledge, they don't embody the knowledge. Users of the methods must bring with them quality attribute expertise and experience.

Our ultimate objective is to have a methodical basis for architecture design that can also be used for architectural analysis. This basis should be grounded in an understanding of how architectural design decisions relate to the achievement of the requirements for the system. Quality attribute models provide the framework within which a methodical design process can exist, but there are two problems with using this framework:

1. It takes a great deal of expertise to analyze quality attribute models. Analyzing models for multiple attributes requires an expertise that few architects have.
2. Managing the interaction among different quality attributes during a design process involves a great deal of iteration to determine the effect a change to improve one attribute (for example, modifiability) will have on another attribute (performance, for example).

For both of these reasons, developing tool support for the process of coupling quality attribute requirements to solutions is very appealing. An expert system to support the portion of the design process that matches quality requirements to solutions would reduce the level of expertise the architect needs to verify a design and would also provide automated support for the iterations.

Two other efforts—one research, one production—exist that are attempting to provide expert assistance for the design process.

- Waypointer is a commercial tool developed by the Jaczone Corporation that provides expert assistance for the Rational Unified Process. At this point, assistance in the design area is limited to detecting inconsistencies in specified designs from a syntactic perspective. The company has an interest in our work and is a possible avenue for transitioning it.
- A group at the University of Tandil[2] is also pursuing expert system support for the design process. Their approach differs from ours in two fundamental ways. First, they visualize the "planner" of the design assistant as being more flexible than we do. Our planner assumes a fixed set of high-level goals, whereas the goals of their planner are more dynamic in nature. And second, we base our composition on the responsibilities the designed system must achieve; they base theirs on the composition of partial design fragments.

---

[2] J. Andrés Díaz Pace and Marcelo R. Campo, "DesignBots: Towards a Planning-Based Approach for the Exploration of Architectural Design Alternatives." To appear in *Proceedings ASSE 2003* (Argentine Symposium on Software Engineering), 32 JAIIO - Jornadas Argentinas de Informaticae Investigacion Operativa. Buenos Aires, Argentina, September 2003.

3.3 Approach

Currently, software architecture design proceeds based primarily on the knowledge of the architect. Various processes exist that support design but they all are meant to organize the architect's knowledge and experience rather than provide guidance on the actual decisions that are being made. A methodical approach to software architecture design that exploited the fundamental rationale behind design decisions would enable systems built from those decisions to behave more predictably. This would have a major impact on the utility of systems as well as their acceptance by customers.

Previous attempts to methodically derive design from requirements were based on transforming functional requirements into the design. See work by Tekinerdogan [Tekinerdogan 00] for a summary and survey of the design approaches. Our insight is that it is the quality attribute requirements that drive design and not the functional requirements. Given this insight, there are four problems to be solved in order to methodically derive design:

1. Characterize the quality attribute requirements.
2. Enumerate the decisions that architects make to achieve quality attribute requirements.
3. Identify a method that moves from individual requirements to specific design decisions.
4. Develop a design method that exploits knowledge of this methodical approach.

Prior to the work supported by Independent Research and Development, we had proposed solutions for the first two problems. Our solution to the characterization of quality attribute requirements we call *quality attribute general scenarios*. Our solution to the decisions that architects make to achieve quality attribute requirements is called *architectural tactics*. These solutions are included in the second edition of the book *Software Architecture in Practice* [Bass 03]. We had also begun work on the third problem. This work is reported in *Deriving Architectural Tactics: A Step Toward Methodical Architectural Design* [Bachman 03].

Our basic approach is that there exists a collection of quality attribute reasoning frameworks that enable a methodical path from quality attribute requirements to architectural design decisions. In fact, it is these reasoning frameworks that provide the rationale as to why the decisions that architects make do achieve the desired requirements.

We have been exploring two reasoning frameworks. One, a fixed priority scheduling framework, is very mature and has been used to understand and enable the achievement of deadline-driven requirements for many years. The other, a modifiability framework, is very ill-formed, but we are successfully deriving some of its elements for our purposes. These two reasoning frameworks, when considered together, provide insight into the characteristics of the other reasoning frameworks that must be bundled together to achieve our goal of methodical design.

One thing that becomes immediately apparent when considering the different types of reasoning frameworks, the number of quality attribute requirements that exist even in small systems, and the potential for interaction among the quality attribute requirements, is that a methodical approach requires tool support. Expecting an architect to keep track of all of the interactions and to have the requisite quality attribute knowledge is not realistic.

This frames the problem we attacked in this IR&D effort. We need to expand our theoretical base to be able to derive design decisions methodically from quality attribute requirements and we need to understand what functions a designer's assistant should provide to support the design process.

3.4 Collaborators

Members of the technical staff at the SEI involved in this project are Felix Bachmann, Len Bass, and Mark Klein.

We are collaborating with Robert Bosch USA Research and Technology Center to implement our tool design. This collaboration takes the form of Bosch developing (with their own funding) an implementation of our tool design. Bosch wishes to rapidly produce designs and prototypes from a set of specifications. They see the architecture design assistant as key to being able to speedily develop an architecture design with personnel who are not experts in the various quality attributes.

(We investigated a variety of possible collaborations. Other collaborations may still materialize but were not relevant in pursuing the work to this point.)

3.5 Evaluation Criteria

Thus far, the evaluation of the work has been through peer reviews. The Bosch collaboration has a more stringent evaluation criterion, however, because they have a set of specifications for an actual product that they are currently producing. The results of the design produced with our tool will be compared with the design produced through the current process. This evaluation will occur in 2004.

3.6 Results

Within the project, we accomplished two major results:

1. The theoretical basis of a methodical approach was clarified to the extent that the designer's assistant can be designed.

2. A preliminary design for a designer's assistant, including platform decisions, has been created.

3.6.1 Clarifying the Theoretical Background

Previously, we knew that the quality attribute reasoning framework provided the path that enabled us to move from requirements to decision decisions, but we were not clear on the exact steps and the modifiability reasoning framework was not sufficiently well understood to provide a sufficient base for ensuring that modifiability requirements were achieved. We have made progress on both of these problems. Furthermore, we had previously only considered single requirements in isolation; we now can deal with multiple requirements, both within a single quality attribute and across quality attributes. More theoretical details can be found in work by Bachman et al. [Bachman 03b].

Deriving Design Decisions Based on Quality Attribute Requirements

There are five steps for moving from quality attribute requirements to design decisions.

1. Specify the quality attribute requirement according to a defined template.
2. Determine the quality attribute reasoning framework that is relevant for that requirement.
3. Bind the parameters of a model based on that reasoning framework.
4. Solve the model to determine if the requirement is achieved.
5. If not, adjust the parameters based on the architectural tactics.

Repeat steps 4 and 5 until either the requirement is achieved or all tactics have been applied and, consequently, the requirement is not achievable.

Expanding the Modifiability Reasoning Framework

The essence of modifiability evaluation is cost. It is the cost of a modification, given a particular proposed design, that determines the extent to which a modifiability requirement will be satisfied. Current cost models are based on system level parameters and not on design parameters. As part of the design work for the architecture assistant tool, we discovered many of the parameters of such a cost model. The model needs to know the dependencies among various modules and the subsequent cost of implementing the ripple effect of changes, and it needs to know the interactions among responsibilities within a single module in order to determine the cost of implementing changes resulting from side effects.

We also defined a procedure for moving from multiple quality attribute scenarios to design decisions to satisfy the requirements of the scenarios.

First the quality attributes are categorized based on reasoning frameworks. Scenarios that can be treated within the same reasoning framework are placed into the same category.

The steps we enumerated above for moving from a single scenario to design decisions also apply to multiple scenarios within the same reasoning framework. These steps are intended to determine the parameters for a model based on multiple responsibilities, and there is no difference whether those responsibilities come from one or multiple scenarios.

Scenarios that cross multiple reasoning frameworks are solved by iterating among the reasoning frameworks. The interaction among the reasoning frameworks consists of responsibilities that are introduced by a reasoning framework (instead of being derived from a scenario). These new responsibilities must be annotated with parameters from all reasoning frameworks. Iteration among the reasoning frameworks will enable this to happen.

Details of this procedure are given in *Preliminary Design of ArchE: a Software Architecture Design Assistant* [Bachman 03b].

3.6.2 Preliminary Design of the Design Assistant

Figure 1 shows a module decomposition view of the design assistant.

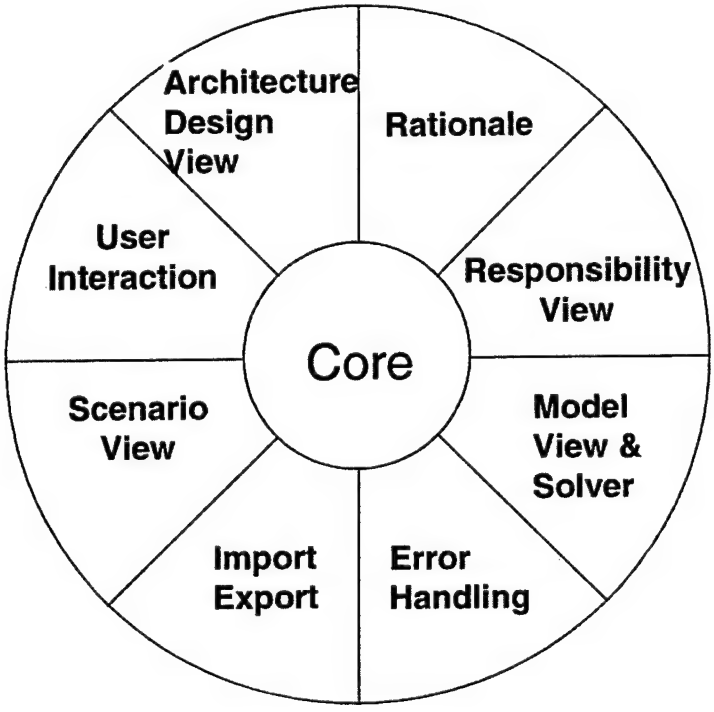


Figure 1: Module Decomposition View of Designer's Assistant

We now describe briefly the function of each of the modules presented.

- Core. This is a set of rules for the rule engine. It manages all of the facts associated with the current design and also the flow of control issues associated with the other modules.

- Scenario View. This allows the input and output of scenarios by the designer.
- Import Export. This manages all import and export of the facts in the fact base.
- Error Handling. This manages any errors that occur during the design process.
- Model View and Solver. This displays the model and current set of parameters for the designer, allows the designer to specify some or all of them, applies tactics to modify the parameters, and solves the result. There will be one instance of this for each reasoning framework.
- Responsibility View. This displays the current set of responsibilities to the designer and allows for modification and specification of the relationships among them.
- Rationale. This allows for the designer to ask questions about the current design, such as what were the values created by the model solver, and how close did the assistant come to meeting particular requirements. We expect this module to be the last implemented.
- Architecture Design View. This allows the designer to see the current design that the assistant has created and to specify portions of the design.
- User Interaction. This handles overall user interaction not covered by the other modules.

3.7 Publications and Presentations

Two SEI technical reports were published as a result of this investigation. They are

- Bachmann, Felix; Bass, Len; & Klein, Mark. *Deriving Architectural Tactics: A Step Toward Methodical Architectural Design* (CMU/SEI-2003-TR-004). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr004.html>.
- Bachmann, Felix; Bass, Len; & Klein, Mark. *Preliminary Design of ArchE: a Software Architecture Design Assistant* (CMU/SEI-2003-TR-021). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

In addition one invited lecture to the Chicago Software Process Improvement Network presented results based on this project. This lecture, "*Software Architecture as a Realization of Business Goals,*" can be found at http://www.sei.cmu.edu/ata/spin03/spin03_1.htm.

4 A Model-Based Reference Architecture (MRA) for Mobile Robotics Systems

4.1 Purpose

Reference architectures have been used with various degrees of success in managing the evolution of systems and the development of product lines. Successful use requires that the reference architecture provide insight into key quality attributes of performance-critical aspects of such systems, and into understanding their tradeoffs and sensitivity points in the architecture.

The purpose of this project is to investigate the feasibility of developing model-based reference architectures that characterize and predict performance-critical quality attributes of software-intensive mission-critical systems, such as timing, performance, reliability, and safety. In order to achieve such a prediction, the following elements need to be in place:

- Commonality and variability of a class of application systems needs to be understood.
- Performance-critical characteristics of such systems must be captured in architecture models as the basis of predictive analysis.
- The model-based reference architecture must be a practical basis for implementation of realistic systems that satisfy the predictions.

The first element leverages the work in Architecture Tradeoff Analysis (ATA). The second element builds on work in Performance-Critical Systems (PCS) and the Carnegie Mellon University (CMU) Electrical and Computer Engineering (ECE) Department, and feeds into ATA and Predictable Assembly from Certifiable Components (PACC) work. The third element builds on current development practices in place at CMU's Robotics Institute and applies technology developed by ECE under the DARPA (Defense Advanced Research Projects Agency) MoBIES (Model-Based Integration of Embedded Software) program.

Creating a model-based reference architecture for mobile robotics systems has the following benefits:

- codification of architecture design and analysis for a family of performance-critical systems in the robotics domain from both the perspective of the application domain and the perspective of a distributed, dependable, real-time software system architecture

- creation of a vehicle for rapid development and evolution of system instances based on the reference architecture through tool-supported architecture analysis and system generation
- development of a method allowing model-based reference architectures to be established in other domains of performance-critical systems

4.2 Background

Many organizations are recognizing that their software-based systems have a great degree of commonality and should be viewed as a family of systems or product line. Similarly, systems are becoming components in systems of systems and those systems must have an architectural infrastructure in place to facilitate integration and interoperation. A key to understanding and managing the evolution of such systems is the provision of a reference architecture.

Reference architectures have taken many forms. They present the system architecture in terms of layers (e.g., open systems interconnection [OSI]), and decomposition into subsystems and services. The system may be viewed as consisting of a physical system, e.g., a tank, and a software system consisting of the application software and the computing platform. Reference architectures are intended to capture commonality and variability in the systems being characterized [Perry 98]. They focus on characterization of interfaces using frameworks such as the SAE (Society of Automotive Engineers) Generic Open Architecture (GOA) framework (see http://www.sae.org/technicalcommittees/aasd.htm) and selection of standards to achieve portability, openness, and interoperability.

For mission-critical systems it is essential to also address key quality attributes such as timing, performance, reliability and safety, which are critical to the performance of such systems, as has been successfully demonstrated by CelsiusTech [Brownsword 96]. These performance-critical quality attributes must be expressed in terms of the application domain (e.g., tolerance of controllers to faulty sensor data), as well as in terms of the software system architecture (e.g., latency in producing output to actuators), which must be consistent with the phase delay designed into the control algorithm.

To address these performance-critical aspects of software-intensive systems, the practitioner community has developed standards representing architecture frameworks specific to time-critical and reliable performance (e.g., Avionics ARINC 653, and automotive OSEK [``Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug" (open systems and the corresponding interfaces for automotive electronics)] Time Triggered Architecture), standards that characterize interfaces to time-critical and reliable services (e.g., Real-time POSIX [Portable Operating System Interface], OSEK Real-time OS and Communication Application Programming Interface [API]), and standards that support architectural modeling of systems with time-critical and dependability characteristics (e.g., OSEK Implementation Language [OIL], SAE Avionic Architecture Description Language [AADL], and Real-

time Unified Modeling Language [UML]). The Object Management Group (OMG) recently has begun promoting Model Driven Architecture (MDA) as a methodological approach to UML-based large scale system development.

Recent advances in architecture modeling and analysis allow us to take a fresh look at how to best capture essential characteristics of reference architectures, architecture choices and decisions that reflect commonality and variability in systems, and tradeoffs and sensitivity points in the system architecture that greatly affect key performance-critical quality attributes. Model-based analysis of such reference architectures can be leveraged when designing particular systems based on a reference architecture, and the architecture model can be the basis for system generation and evolution.

4.3 Approach

Our approach to developing a model-based reference architecture consists of three phases: architecture discovery, architecture capture and model-based analysis, and reference architecture validation.

4.3.1 Phase I: Architecture Discovery

During phase I we worked with the Robotics Institute at CMU to evaluate the architectures of several existing autonomous vehicle systems (we anticipate evaluating three candidates). To perform the evaluation we used a miniature variant of the Architecture Tradeoff Analysis Method[SM] (ATAM[SM]). At the conclusion of the ATAM evaluations, we identified commonalities in the architectures such as

- common driving quality attributes
- reoccurring software architectural patterns in the systems
- common architectural mechanisms
- common components and connectors
- reoccurring risk themes resulting from architectural decisions

4.3.2 Phase II: Architecture Capture and Model-based Analysis

Based on our findings from the ATAM evaluations, we have been creating a reference architecture by applying architectural models and modeling notation to support multiple architectural views. The reference architecture includes traditional documentation associated with

[SM] Architecture Tradeoff Analysis Method and ATAM are service marks of Carnegie Mellon University.

software architectures such as components and connectors and their topology, as well as key quality attribute considerations.

In addition to traditional architectural documentation, we are using architecture modeling notations to capture key aspects of the reference architecture for analysis with respect to various quality attributes common to autonomous systems, such as timing (in terms of control systems and in terms of software tasks), performance, and safety, and for system generation.

4.3.3 Phase III: Reference Architecture Validation

Once the reference architecture has been established, we will validate that it provides value to system designers.

- We will perform a model-based analysis of the reference architecture model with respect to those quality attributes that are key drivers, using model-based architecture analysis tools.
- We will identify how model-based analysis results can support an engineer's intuition about design decisions made in system architectures.
- We will port one of the candidate systems into the model-based reference architecture, analyze it and compare the results with the actual system. If feasible, we will regenerate the implementation for one configuration through an architecture-driven system build tool.

The IR&D project exercises these phases on autonomous robotics systems. However, due to the scope of the effort, part of the phase III activity will be carried out in a follow-on activity.

4.4 Collaborators

We are testing this approach by analyzing various autonomous robotics systems being developed by the Robotics Institute to discover common architecture patterns, relevant quality attributes, mechanisms used successfully, and implementation issues. We will then consolidate our findings into a reference architecture for mobile robotics systems with an architecture reference model, analysis models, and guidance and heuristics for building similar systems.

We will leverage the collective work of the SEI in the area of software architecture; the ECE Department's work in robotics and distributed real-time systems; and the Robotics Institute's work in several areas of autonomous robotics systems, covering the spectrum of a global five-layer robotics architecture to create a model-based reference architecture for autonomous systems.

Peter Feiler of the PCS initiative is the co-author and editor of the SAE Avionics Systems Division (ASD) Avionics (Embedded Systems) Architecture Description Language (AADL) standard. This standard addresses the need for modeling of large-scale embedded systems,

predictable analysis of their task and interaction architecture with respect to performance-critical run-time attributes, and automated architecture-driven system integration. It will go into ballot at the end of 2003. The AADL is extensible to accommodate a range of architecture analysis techniques. Tony Lattanze of the ATA initiative is intimately familiar with ATAM and brings a wealth of experience, technology, and methodology from ATA to the table.

Dio de Niz and Raj Rajkumar are from the ECE and CS departments at CMU. They are active in the area of embedded real-time systems. Raj Rajkumar is an expert in scheduling analysis and quality-of-service-based resource management and leads efforts in real-time operating systems and real-time multimedia applications. Dio de Niz has developed a model-based framework (TimeWeaver) that separates the encoding of functional and parafunctional behaviors into semantic dimensions (e.g., event flow, timing, concurrency, fault-tolerance, deployment) that can be modified independently. The impacts of changes in one dimension on the realization of other dimensions are automatically projected and managed. Platform dependencies are also captured separately, enabling a code-generation subsystem to reuse the same components across a wide range of heterogeneous platforms and applications. System components can be recursively composed or decomposed. An analyzable software structure is enforced such that the end-to-end timing behavior of the resulting system can be verified.

Jay Goudy and Reid Simmons from the Robotics Institute are bringing to the table robotics systems and the methodology and technology in actual use in such systems. One family of candidate systems, environmental systems to support intelligent busses, is under the responsibility of Jay Goudy. This family represents the sense and understand layers of a global multi-layered robotics domain architecture. Like most robotics systems, it takes a modularized approach with adaptable interfaces, is mostly signal stream driven, and uses a distributed shared memory communication architecture. Reid Simmons is the research scientist responsible for a second class of candidate systems (exemplified by social robots such as GRACE, NASA space probes, and multi-robot systems for planetary exploration): autonomous robots with fixed missions that cover the next two layers of the global robotics architecture. Their implementation architectures tend to be message-based, using publish/subscribe—focusing on task execution and multi-task planning, while the lower layers are lumped into a single process system component.

4.5 Evaluation Criteria

For the MRA effort, we will consider the following evaluation criteria:

- an initial reference architecture and methodology for using the reference architecture as a basis for analysis of robotics systems
- completed analysis and reconstruction of one of the existing candidate systems

- the system architects from the Robotics Institute benefiting from the model-based analysis and continuing to use it in future projects
- cross-fertilization between different groups at the Robotics Institute by leveraging software system implementation architectures and supporting technologies and the analyzed reference architecture

In the future the results of this IR&D project will enable system designers and architects of autonomous systems to use heuristics, tools, and analysis methods (where possible) to make design decisions. This will help developers build systems with more predictable properties in less time. Ideally the products of this IR&D effort could be used in collaboration with an SEI client to build an autonomous system. If we are able to sufficiently codify this information, collaboration with a commercial company may be possible and may enable the construction of tools to further help designers and developers.

4.6 Results

The MRA IR&D is still in progress at the time of this writing, so these are only preliminary results. This section summarizes the results of the architecture discovery phase and the architecture capture and analysis phase.

4.6.1 Architecture Discovery

We have examined two classes of robotics systems: environmental systems and autonomous systems.

Environmental Systems

The environmental systems group builds systems that perceive the environment and provide feedback to another system or user. These systems are predominantly constructed as loosely coupled modules that perceive the environment and then perform arbitration, control, and planning tasks. Information from these systems can be used by humans in the form of alarms and advisory information, or it can be used by other systems to control their behavior.

The driving quality attributes for the Environmental Systems group are modifiability and composability. System components are often built in isolation, to be integrated into a system when ready. As a result the systems architect and engineers operate with the following assumptions:

- There are no pervasive, permanent standards—so don't assume that there will be.
- Modules are code units that are isolated from the architecture in which they reside.
- Modules have visibility into the system in which they reside through reconfigurable interfaces that are standard only to that module.

- Reconfigurable interfaces are used to hide adapters that integrate modules into a particular architecture.
- Tools are used which can select and reconfigure module adapters at run time.

This has led to systems built by this group being largely based on the component connector paradigm. At run time the connector used is distributed shared memory. As the name implies, processes need not be collocated on a single processor to share memory. The details of shared memory access are hidden in a standard interface.

These approaches and the associated technical framework used by the environmental system group have provided numerous benefits, but there are also a few issues as well.

The first issue seems to be related to composability. While modules are easily built, it seems that assembling them into a useful system is more of a challenge than it should be. It clearly indicates a need for support in predictability of composed systems.

It is tempting to assume that systems like those built by the environmental system group have stringent availability, performance, and safety constraints, but this is not the case.

For the most part, there is no formal scheduling analysis (e.g., rate monotonic analysis). Processes run as fast as they can and adjustments are made in the system to accommodate any resource limitations. In essence the systems are viewed as soft real time, although they do have hard real-time characteristics, but can afford to occasionally miss deadlines due to the fault tolerance of control systems to handle transient lack of data from sensors.

Extraordinarily high levels of availability are not a priority for environmental systems either. Systems built by this group are largely prototypes to support research or demonstrators for potential applications.

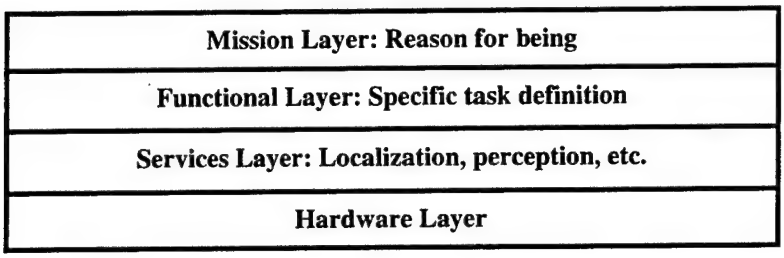
Safety is not a primary concern for this group. Many of their systems are man-in-the-loop systems. When a fault does occur in one of their systems, it is sufficient to report the fault.

One of the major stakeholders indicated that one of the "horrors" of system development is that, even with the technical framework, "module developers must know details of the system they are building and system integrators must know the details of how modules work."

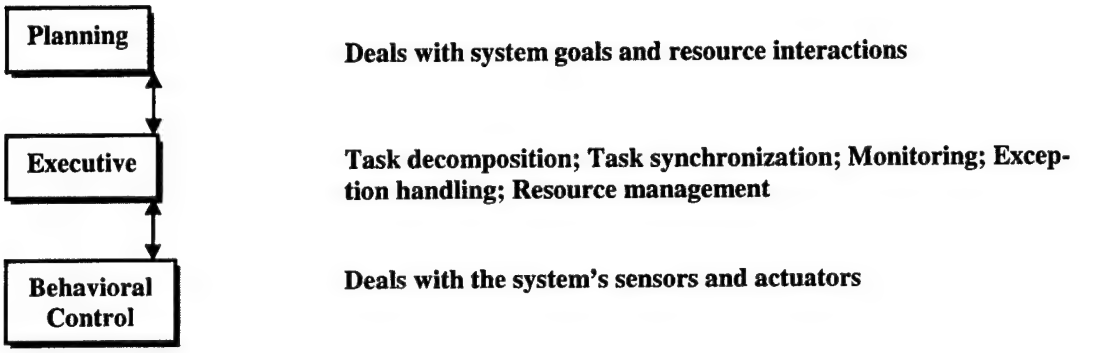
Autonomous Systems

From a run-time perspective, the autonomous systems use an architectural style of communicating concurrent processes. The Autonomous Systems Group uses a publish/subscribe mechanism that functions across processor boundaries and shared memory.

While these systems differ significantly from one another, the Autonomous Systems Group often abstracts the behavior of autonomous systems in terms of a simple layered diagram:



Another layering approach used by the Autonomous Systems Group is a three-tiered architecture. This should not be confused with three-tiered architectures common in the IT domain, but is a special abstraction that is used to model robotic systems. The general three-tiered architecture is similar in some respects to the previous "layered" model:



Modifiability is a major architectural driver for the Autonomous Systems Group as well. A primary motivation for promoting modifiability in these systems is to achieve a high level of reuse. While this is a driving goal, most robotic systems are built from scratch.

The Autonomous Systems Group has a similar philosophy as the Environmental Systems Group regarding performance, availability, and safety. With respect to performance, the philosophy is that tasks run as fast as they can, given the bandwidth and communication requirements between systems and system components. No scheduling analysis is performed on the systems. Hard, predictable, deterministic response time is not required.

However, the systems have had performance issues. A stakeholder indicated that systems have "slowed down in non-deterministic ways, causing problems" and explained that these are usually difficult and time-consuming problems to find.

High levels of availability and safety are not a priority for these systems. Systems built by this group are largely prototypes to support research or demonstrators for potential applications.

In summary, the systems that the Robotics Institute builds are research systems, thus they do not necessarily reflect the needs of production quality autonomous robotics systems. Their developers think they do not have the performance, availability, and safety concerns that production, mission-critical systems have. However, they do encounter operational problems that

are rooted in the fact that performance and reliability analyses have not been performed. The implementation architectures address the need for modifiability and configurability as well. Some of the operational techniques for model-based analysis of system behavior through model checking are currently being investigated by RI.

4.6.2 Architecture Capture and Model-Based Analysis

The Robotics Institute is developing a Task Description Language (TDL) model for the planning and execution of autonomous robot tasks. We have examined the notation and example models and compared it to the emerging AADL standard.

The TDL has very complex syntax and semantics, due to the fact that its original goal was to simplify the use of a particular set of run-time libraries. The notation has exclusive focus on control flow–leaving data flow embedded in the application code–typically done through global data. The implementation of TDL is an extension of C++ and is handled as a pre-processor.

The comparison with AADL identified improvements to the TDL by encapsulating certain common patterns of TDL description into a more abstract construct that is directly supported in the AADL as a modeling concept. Much of the architecture can be modeled with the AADL, including dynamic behavior modeled through the mode concept.

The need for dynamic planning and execution resulted in highly complex constructs in TDL that allow the developer full control over when tasks are planned and executed. Of concern is the fact that planning takes effort and alternate plans must be considered as the autonomous system encounters situations not addressed by the original task sequence. This problem can be abstracted into the architecture model as a resource scheduling problem. A solution to this problem is currently being developed, which will result in a great simplification in TDL, and a possible inclusion of such planning and plan execution scheduling analysis into AADL.

We are also in the process of capturing one of the autonomous system architectures in TimeWeaver. A TimeWeaver-based model allows us to demonstrate several forms of architecture analysis, such as scheduling analysis and addressing reliability through redundancy. Furthermore, it is also the basis for the validation of a reference architecture through reconstruction of the system from an architecture model.

4.7 Publications and Presentations

"Summary of Robotics Architecture Observations." Presentation and working draft.

"Task Description Language: An Analysis and Comparison with SAE AADL." Presentation to project group.

5 Securing Wireless Devices

5.1 Purpose

The world is going wireless. The U.S Department of Defense (DoD) view of the future battlefield includes a wide variety of wireless support at all levels, from the command post to the foot soldier in the field. Our experience in the civilian wireless arena indicates that this is fraught with peril.

The purpose of this project was to develop a research program that leverages the prior experience of the investigators to secure both personal wireless devices and elements of an autonomous wireless support infrastructure.

5.2 Background

Over the past two years, we have become increasingly aware of a variety of problems with the security of wireless devices, both inside and outside the DoD:

- A 2001 Defense Information Systems Agency workshop on malicious code raised questions about the security of wireless personal digital assistants (PDAs) used by military officers. (Principal investigator McHugh was a participant.)
- Small, portable devices are subject to loss and capture, as are unmanned relay points. (Collaborator Reiter has worked on countermeasures.)
- CERT staff found widespread wireless vulnerabilities during their support of the U.S. Secret Service (USSS) at the 2002 Olympic Winter Games in Salt Lake City. (McHugh advised on equipment.)
- Authentication of operators is a problem with these devices. (Reiter has worked on biometric approaches for enhancing authentication.)
- The National Academies' oversight panels for the Army Research Labs have found widespread problems with current and proposed battlefield wireless systems. (McHugh is a member.)
- The tendency toward adoption of commercial off-the shelf (COTS) solutions makes it likely that civilian technologies (cell-phone-based, 802.11, etc.) will be used by the DoD

despite vulnerabilities. (McHugh gives presentations to DoD and civilian groups on the risks.)

- Wireless security is recognized as a high-priority item in the Working Group report on challenge problems and in the IR&D call for proposals.

The SEI customer community has a serious interest in wireless communications and recognizes that security is a major concern.

5.3 Collaborators

The following people participated in this project:

- SEI members of the technical staff (MTS) and principal investigators: John McHugh, Sven Dietrich
- Other CMU: Mike Reiter, Professor in Electrical and Computer Engineering (ECE)

5.4 Results

By the success criteria set forth in our proposal submitted in July 2002, this effort must be judged as a failure. We have discovered no suitable funding opportunity. No research prototypes have been produced and no papers have been written for submission to journals or conferences.

On the positive side, there have been a small number of minor activities associated with the project that might be viewed in a favorable light:

- A wireless information page was established on the Internet Security Alliance Web site (http://www.isalliance.org/knowledgebase2/knowledgebase.htm) This site was maintained by ECE undergraduate Gerald Zhou during the spring and summer of 2003.
- We have continued to make presentations to customer and other groups, including the Industrial Affiliates of the ECE Center for Wireless and Broadband Networking, on the risks of wireless networking.
- Chris Bateman of the CERT Analysis Center (in an effort not supported by the IR&D funding) is working with the USSS in the area of wireless monitoring. While we took part in the preliminary discussions for this project, we did not directly participate in the effort and cannot attribute its existence to the IR&D funding. The project has been developing an Adversarial Wireless Assessment Study, which, according to its statement of work, "will provide an overview of the wireless/portable computing threat environment. From the results it is hoped that the USSS will be able to more clearly focus its efforts, provide significant assistance to other law enforcement entities, and better ensure the security and survivability of its own, developing wireless capability." When the study is

finished, it will be given a closely controlled distribution, as it reflects on the operational capabilities of the Secret Service.

5.5 Lessons Learned

There are a number of reasons for this failure and it is worth discussing them as a cautionary note for future endeavors.

- Unfunded collaborations are problematic.

  Mike Reiter has continued work on capture-resistant systems, working with a student, and has produced a paper that will appear later this month, but the collaboration that we had hoped for has not materialized.[3] Early in the project, we managed to schedule a few regular meetings, but the press of other commitments for all of us resulted in their abandonment after a short time. This is in contrast to our experience the previous summer when CERT supported Mike in connection with our project on Active Network Defenses.

- The gulch factor.

  It has been our experience that many of the benefits of collaboration are lost when the working conditions are such that chance encounters between colleagues do not occur. We had hoped that the founding of The Center for Computer & Communications Security (C3S) would create an environment in which faculty and students working in the security area would be in frequent and close contact. Although we have an office in the C3S space, we seldom use it because none of the faculty or graduate students involved in security research have a presence there. Whenever possible, we drop in on colleagues on campus when we go there for seminars, research talks, etc., but this is strictly one-way traffic. The combination of the 15-minute walk and the intimidating security precautions used at the SEI seem to preclude drop-in visits from colleagues or students from campus.

- Inappropriate choices of target.

  In retrospect, we see that many of the things that we had hoped to do under the project require facilities and support not available at the SEI. We had hoped to explore the integration of positioning services such as GPS with wireless devices to support location-dependent security. Much of what we had hoped to do requires a modestly equipped laboratory and test equipment. Although we could have purchased some of the test equipment under the program, we saw no support for establishing a hardware-based laboratory in the long run and abandoned the effort after purchasing a limited number of wireless cards, access points, and antennas. Some of the things that we would like to do require a substantial programming effort. Our own code-writing skills are somewhat ob-

[3] Reiter, M. K.; Samar, A.; & Wang, C. "The Design and Implementation of a JCA-Compliant Capture Protection Infrastructure" (extended abstract). To be published in *Proceedings of the 22nd IEEE Symposium on Reliable Distributed Systems*, October 2003.

solete and there is no pool of low-level (device driver, etc.) programmers on which to draw.

- The press of other activities.

  As planned, the project did not get started until January of this year due to prior commitments by both CERT investigators. During March and April, several major funding opportunities through Advanced Research and Development Activity (ARDA) and Defense Advanced Research Projects Agency (DARPA) materialized. These took the form of Broad Area Announcements that explicitly encouraged the competitive participation of the SEI (ARDA) or provided vehicles for submission of a white paper in lieu of a proposal. In addition, we were able to team with CMU through C3S in preparing proposals in response to NSF's midscale ITR. A substantial amount of time was spent in preparing proposals and/or white papers in support of these and related efforts. The proposal preparation activity preempted the IR&D efforts, which were stalling by that time, and momentum was never regained. During the summer, a substantial portion of John McHugh's time has been devoted to activities for our ARDA customer, who seems to be a potential source for additional funding. Unfortunately, this has involved several week-long trips to San Diego.

  As of the present time, we have been fairly successful with the proposals. One ITR submitted by Chenxi Wang has been funded by NSF, and Sven Dietrich and John McHugh are supported at the 8 percent each level for the next three years and at the 5 percent level for two additional years. McHugh teamed with Orincon on an ARDA proposal, which has been selected for an award. This will cover about 10 percent of his time for three years. The white paper that we sent to DARPA in connection with the Dynamic Quarantine program has been accepted. This gives about $950K over 18 months (with an opportunity for an additional 18 months if our initial efforts are successful), and was originally intended to provide 50 percent support for Dietrich and McHugh, as well as substantial support for others. Unfortunately, DARPA has decided to do the project at the SECRET level, which precludes Sven's involvement (Sven is a German national), but he has developed other sources of funding. Several other ARDA and National Science Foundation (NSF) proposals are still pending, and an additional white paper to DARPA is in preparation.

5.6 Conclusion

In summary, it appears that much of what we set out to do is not really appropriate research for an organization such as the SEI. The support effort for the USSS is much more in line with the CERT's mission and methods of operation, but it is not research in the sense that we would like. When we look at research opportunities in this area, we see that relatively few of them fit the SEI model. For example, much of the future of wireless security is in the hands of the standards groups that are defining the protocol features. Participation in these groups is voluntary and participants are expected to pay their own way. Effective involvement would

probably require a fairly extensive evaluation and development facility as well. We do not think that SEI customers are likely to sponsor our involvement in these activities on an ongoing basis. Emerging technologies such as Ultra Wide Band (UWB) are in their formative phases, but much of the potential for security research in this technology lies with the hardware developers. While it is likely that UWB will prove to be no more secure than its predecessors, we see no leverage point that we could use to affect its development. We will continue to participate in the Center for Wireless and Broadband Networking and hope that the SEI may be better positioned in the future. For now, we feel that our research efforts will be more successful if we concentrate on areas in which we can make contributions without substantial outlays for laboratory facilities.

6 Sustainment

6.1 Purpose

Motivation for the Sustainment IR&D project came from both the core statement of the SEI mission and the DoD Challenge Problems.

The core of the SEI mission is to provide "...technical leadership to advance the practice of software engineering so the DoD can acquire and *sustain* its software-intensive systems with predictable and improved cost, schedule, and quality."[4] There has been little emphasis on sustainment of software-intensive systems since the SEI was founded in 1986. The intent of this IR&D study was to begin to explore and understand software system sustainment in the DoD and supporting software industry and to seek ways to quantify the sustainability of existing systems. Measuring sustainability is a necessary prerequisite to the future validation of any processes, techniques, or tools for improving system sustainability.

Sustainment is one of seven SEI thrust areas outlined in the 2002 SEI Strategic Plan. As stated in "Software Engineering Thrusts and Challenge Problems":

> *The sustainment phase of a system must be considered in software acquisition, but currently software engineering provides little guidance for explicitly and credibly reflecting future considerations and value into early system analysis, design, and architecture....Software engineers and managers alike must confront the task of understanding the potential for tailoring a legacy system for new mission needs and the costs of doing so. Today's tools for assisting in this task are woefully inadequate.*

The sustainment roadmap in the strategic plan defines a multiyear plan for the SEI to address this challenge area.

[4] Emphasis added by the authors.

6.2 Background

6.2.1 Legacy System Crisis

The amount of code in legacy systems is an indicator of the criticality of sustainment. It is immense and growing. This became evident in the late 1990s, when organizations hired legions of COBOL programmers and consultants to fix Y2K-related problems. In 1990, it was estimated that there were 120 billion lines of source code (primarily COBOL and FORTRAN) in existence [Ulrich 90]. Since 1990, there has been a huge increase in the use of computers for business process support. One estimate is that roughly 250 billion lines of source code must now be maintained and this number is increasing all the time [Sommerville 00]. The average Fortune 100 company, for example, maintains 35 million lines of code and adds ten percent each year just in enhancements, updates, and other maintenance. As a result, the lines of code that the average Fortune 100 company has to maintain will double in size every seven years [Müller 94]. According to an informal industry poll, 85 to 90 percent of IS budgets goes to legacy system operation and maintenance [Erlikh 00]. More software means having more software to maintain and to evolve to enhance functionality in response to changing business practices; to correct errors; or to improve performance, maintainability, or other quality attributes.

The problem does not stop here. Lehman's second law states: "As an evolving program changes, its structure becomes more complex, unless active efforts are made to avoid this phenomenon" [Lehman 80, Lehman 85]. Increased complexity makes the system more brittle, increases the chance of side effects with every change, and makes future software maintenance more difficult.

The result of these factors is a *legacy crisis*: an ever-increasing source code base for which maintenance becomes increasingly difficult. This crisis can be averted only with a shift in paradigm for software development and acquisition. Rather than the typical life cycle of system development, periodic maintenance upgrades, occasional modernization, and eventual replacement, we must adopt a process of initial construction, followed by *continuous improvement* activities to achieve currency with evolving business needs. System development does not end at deployment, but continues (albeit at a reduced effort) throughout the life of the system.

6.2.2 Current SEI Efforts

Sustainment work has been, and is being, performed in a variety of existing initiatives.

Sustainment work performed by the COTS-Based Systems (CBS) initiative included the development and publication of an SEI Series in Software Engineering book, *Modernizing Legacy Systems*, which chronicles SEI experiences working on the ILS-S and IMDS moderniza-

tion efforts at Gunter AFB and FedEx ground (formerly Roadway Package Systems). CBS also developed a diagnostic programming technique used for discovering information about legacy component interfaces through examination of the components in the context of the original system.

Sustainment work performed by the Integration of Software-Intensive Systems (ISIS) initiative includes establishment of sustainability metrics and work in assumptions management. These ideas were directly influenced by the Sustainment IR&D and are discussed in further detail in the Results section.

Sustainment work in the Product Line Practice (PLP) initiative focuses on the technical aspects of mining software assets, an integrated approach to migration planning, and conceptual understanding of basic reengineering principles. The technical approaches focus on two aspects of mining assets for a product line: (1) mining existing systems at an architectural level through Mining Architectures for Product line (MAP) evaluation,[5] and (2) mining components for a product line through Options Analysis for Reengineering (OAR).[6]

6.3 Approach

Our approach in the execution of the Sustainment IR&D was to build a foundation for further work in software sustainment at the SEI by defining what is meant by a sustainable system and identifying or creating a measure of sustainability. Establishing a measure of sustainability is a necessary prerequisite to validating tools, processes, and techniques aimed at improving the sustainability of an existing system. Management of sustainment teams and organizations is inadequately supported by existing measures, in that these measures cannot be readily applied to improving system sustainability or in evaluation. For example, a common maintainability measure is mean time to repair (MTTR) [Fenton 91]. MTTR may be used, for example, to reduce turn-around times. On the surface this appears to be a worthy goal. However, if measuring MTTR encourages developers to rush fixes, resulting in additional downstream defects, these efforts may actually reduce the overall sustainability of the system. Tracking the rate at which modification requests are submitted is also problematic. A low submission rate may reflect user frustration over an existing backlog of requests that has not been attended to, rather than indicate increased product quality.

MTTR and other existing measures inadequately support evaluation, for example, when selecting between continued sustainment and replacement or deciding which of multiple redundant systems to keep and which to eliminate. The problem in both cases is that existing maintenance measures fail to adequately measure the gap between the existing and desired state of the system.

---

5 http://www.sei.cmu.edu/plp/products_services/map.html

6 http://www.sei.cmu.edu/reengineering/oar_method.html

A common problem with MTTR and other existing measures is that, taken in isolation, the results are inconclusive as to the status of the sustainment activity. Various measures need to be considered in tandem and a hypothesis produced that explains the measures. This hypothesis then needs to be tested and validated. While this process may be successfully applied by an adept and experienced management team, it does not form the basis of a repeatable process. To extend measurement-based process improvement beyond an elite subset of practitioners requires a comprehensive set of measures that can be readily interpreted.

If successful, the work started in the Sustainment IR&D could form the foundation for an initiative in software sustainment by providing a basis for measurable improvement. The establishment of a sustainability assessment tool may also be a useful service that can be offered to SEI customers and provide an entry into broader engagements.

6.3.1 Potential Impact

Sustainment as a percentage of life-cycle costs is high and increasing as shown in Figure 2. As of the early 1990s, sustainment costs were over 90% of overall life-cycle costs. This means that all other software engineering research focused on the initial development effort is trying to optimize an area that represents less than 10% of all life-cycle costs. Focusing on sustainment could have a potentially huge impact by comparison, especially since sustainment is not well researched.

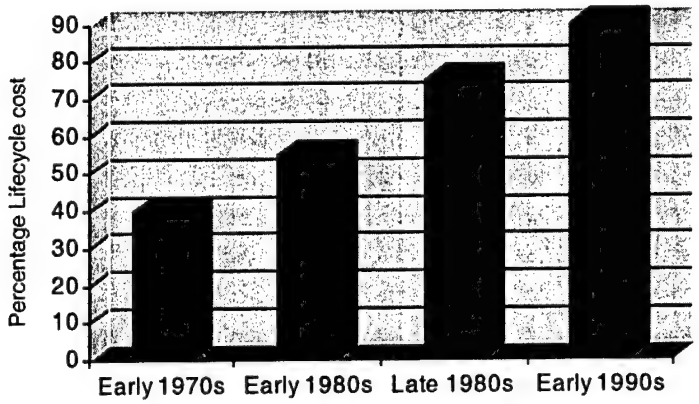


Figure 2: System Life-Cycle Costs

The goal of SEI efforts in sustainment is to help the DoD and industry to

1. develop systems that can be sustained or evolved
2. sustain systems so that they remain sustainable (evolvable systems)
3. modernize existing legacy systems into sustainable systems

6.4 Collaborators

6.4.1 Research Collaborators

The following SEI staff are participating in the Sustainment IR&D project:

Name	Title	Program
Wolf Goethert	Sr. MTS	Software Engineering Process Management
Lutz Wrage	Visiting Scientist	Dynamic Systems
Daniel Plakosh	Sr. MTS	Acquisition Support
John Robert	Sr. MTS	Acquisition Support
Joseph Elm	Sr. MTS	Acquisition Support
Grace Lewis	MTS	Dynamic Systems
Robert C. Seacord (principal investigator)	Sr. MTS	Dynamic Systems

6.4.2 Advisory Board

The Sustainment IR&D project has established the following review board of experts for review and guidance:

- Lt. Col. Jon Dittmer, PM for ILS-S and Chief of the Supply Systems Division at Standard Systems Group (SSG).
- Kenneth Heitkamp, SSG Technical Director/Software Factory Director, USAF
- Ira Baxter, CEO, Semantics Design
- Daniel Yellin, Director, Software Technology, IBM
- Lester Reagan, Deputy Program Manager for ILS-S, USAF
- Edgar Dalrymple, Division Manager, AMCOM, U.S. Army
- Terry Bennet, Vice President, IT, Federal Reserve Bank of Cleveland
- Dennis Smith, Senior MTS, SEI
- Liam O'Brian, Senior MTS, SEI

6.5 Evaluation Criteria

Validation of the measures is a critical issue. Validation of the parameters defined by the IR&D project (e.g., Weighted Modification Request Days, or WMRD) as a measure of sustainability requires repeated confirmation by empirical evidence. If the data supports these

parameters as accurate and reliable measures of sustainability, they can be provisionally accepted as valid, pending further information and tests.

To date, the following concrete steps have been taken to validate our measures:

- WMRD has been the topic of extensive argument and discussion among the authors and between the authors and our external reviewers, and has been considerably evolved through this process.
- The approach has been reviewed and discussed with all the members of the advisory board.
- A paper, "Measuring Software Sustainability," has been accepted at the 2003 International Conference on Software Maintenance, providing additional, external validation of our approach (see Publications section).
- Several sustainment assessments have also been performed, and the collected data is being analyzed to further validate the measure. None of the data collected so far indicates that the measure is invalid for its intended use in measuring the exchange of modification requests and software releases between the sustainment team and the customer.

Further assessments are being performed; when they are complete, the results will be shared with the organizations for which the assessments were performed to further validate the results.

6.6 Results

The Sustainment IR&D was successful in creating a definition of sustainability, in developing measures that can be used to access sustainability, and in performing several sustainability assessments at multiple customer locations in the DoD, government, and industry.

Sustainability was defined by the team as "the ability of a sustainment team to modify a software system based on customer needs and deploy these modifications." This definition is important because it leads directly to how the quality attribute is measured.

The Sustainment IR&D was successful in developing a set of "Historic Measures" and "Change Potential Measures" that can be used to measure the overall sustainability of a software system. These measures are captured in Figure 3.

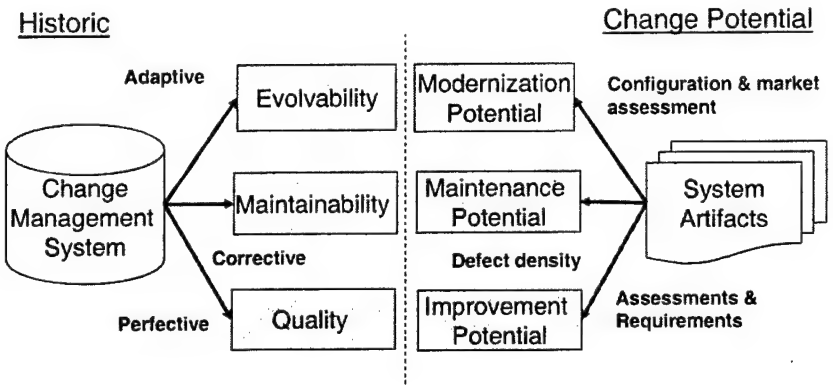


Figure 3: Sustainability Attribute

Historic measures are indirect measures of explicit, expressed needs. A common formula is used to measure the ability of the sustainment team to perform adaptive, corrective, and perfective maintenance processes. One of the measures defined is the "weighted modification requests days" (WMRD) (see Figure 4). This parameter measures three attributes of the adaptation process: the effort required to adapt the software, the time it takes to make the adaptation, and the number of adaptation requests made.

$$\sum_{i=1}^{\#\text{ of open modification requests}} \text{estimated\_change\_effort}_i * (\text{snapshot\_date} - \text{submission\_date}_i)$$

Figure 4: WMRD

Change Potential Measures evaluate the differential between the actual properties of the system and the desired properties of the ideal state. They measure potential modification requests that may be submitted to close the gap between the actual and desired properties of the system. Unlike a predictive measure, change potential focuses on the latent potential for change that exists in a system. While achieving this ideal state is improbable, identifying it defines a standard against which various solutions can be measured.

6.6.1 Sustainability Assessment

The SEI has incorporated the sustainability measures described in this paper into a sustainability assessment tool. An assessment is performed in three steps. First, a questionnaire is completed by the sustainment team. Second, an on-site visit is conducted to correct any deficiencies in the data and agree on a procedure for completing the assessment (including assigning roles and responsibilities). The final analysis of the data is performed by the assessment team. An assessment results in data analysis reports, comparative data, and a guide to interpreting the results. Other data is also provided to the sustainment organization in addition to the six basic sustainability measures described in this report.

Multiple sustainment assessments have been performed to date with additional assessments pending. The data analyzed to date partially validates the measures for their intended use. These measures must be further validated across a broader range of software sustainment efforts before being broadly adopted.

6.7 Publications

Seacord, R. C.; Elm, J.; Goethert, W.; Lewis, G. A.; Plakosh, D.; Robert, J.; Wrage, L.; & Lindvall, M. "Measuring Software Sustainability." To be published in the proceedings of the International Conference on Software Maintenance, September 22-26, 2003, Amsterdam, The Netherlands.

7 System of Systems Interoperability (SOSI)

7.1 Purpose

The importance of interoperability cannot be denied. Interoperability to achieve information superiority is the keystone on which future combat systems (e.g., Air Operations Center, Future Combat Systems), logistic systems (e.g., Global Combat Support System), and other government systems (e.g., interoperability between organizations for homeland security) will be constructed. *Joint Vision 2020* [DSPP 00], which guides the continuing transformation of America's armed forces, states, "Interoperability is the foundation of effective joint, multinational, and interagency operations."

Currently, there is a tendency to concentrate on the mechanisms used by the various systems to interoperate. However, concentrating on mechanisms misses a larger problem. As *Joint Vision 2020* suggests, to create and maintain interoperable systems of systems, there is a need for interoperation not only at the operational level, but also at the level of system construction and program management. Improved interoperation will not happen by accident and will require changes at many levels.

While many systems produced by Department of Defense (DoD) programs can, in fact, interoperate with varying degrees of success, the manner in which this interoperation is achieved is piecemeal. In the worst case, interoperability is achieved by manually entering data produced by one system into another—a time-consuming and error-prone process. Clearly, if America's armed forces are to achieve *Joint Vision 2020*, and if cross-organizational homeland security capabilities are to be developed, a better way forward must be found.

The purpose of this IR&D project was to respond to the need for the SEI to address the issue of interoperability. The study was based on the belief that interoperability must occur at multiple levels between programs and not solely in the context of an operational system. We looked at the full range of barriers—programmatic, constructive, and operational—to achieving interoperability between systems, as well as to candidate solutions to those problems.

7.2 Background

In keeping with Joint Vision 2020, interoperability is receiving increasing and widespread attention. A range of DoD and related organizations are attempting to define the problem, provide solutions, and build interoperable systems. Some of these organizations include

Commands, Directorates and Centers

- Defense Information Systems Agency (DISA) Interoperability Directorate (http://www.itsi.disa.mil/standards.html)
- Interoperability Technology Demonstration Center (ITDC)
- Interoperability Clearing House (ICH) (http://www.ichnet.org)
- Joint Interoperability and Integration Directorate (JI&I) (http://www.teao.saic.com/jfcom/html/charter.html)

Standards and Strategies

- Defense Information Initiative Common Operating Environment (DII/COE) (also called COE) (http://www.dis.anl.gov/is/DIICOE.html)
- Joint Technical Architecture (JTA) (http://jta.disa.mil/)

Demonstrations and Test Beds

- Distributed Engineering Plant (DEP)
- Joint Distributed Engineering Plant (JDEP) (http://jitc.fhu.disa.mil/jdep/)
- Joint Warrior Interoperability Demonstration (JWID) (http://www.jwid.js.mil/)

Joint Force Integration Initiatives

- Blue Force Tracking (BFT) (http://www.fcw.com/fcw/articles/2003/0526/web-blue-05-30-03.asp)
- Combat Identification (CID) (http://www.fas.org/spp/military/docops/defense/actd_mp/CID.htm)
- Family of Interoperable Operational Pictures (FIOP) (http://www.dtic.mil/ndia/systems/Quinlan.pdf)

7.3 Approach

This section includes a discussion of a model of interoperability, which was a starting point for this research, and the method used to conduct the study.

7.3.1 Interoperability Model

There are cases in which systems are interoperating with other systems and the interoperation is achieved through heroic effort and at great expense. Too often, the approaches used lead to interoperability that is specific to the targeted systems (sometimes called "point to point") and do not facilitate extension to other systems. Even then, the technical approaches employed, such as DII/COE and Extensible Markup Language (XML), offer only partial interoperability.

Achieving large-scale interoperation among systems will require a consistently applied set of management, technical, and operational practices that support the addition of new and upgraded systems to a growing interoperability web. Improvements in technology alone (whether XML or any other) will not be sufficient. There must be parallel improvements in identifying current and future interoperability needs, and in how organizations pursue interoperability.

A simple model, shown in Figure 5, depicts the broad range of activities that are necessary to achieve interoperability.

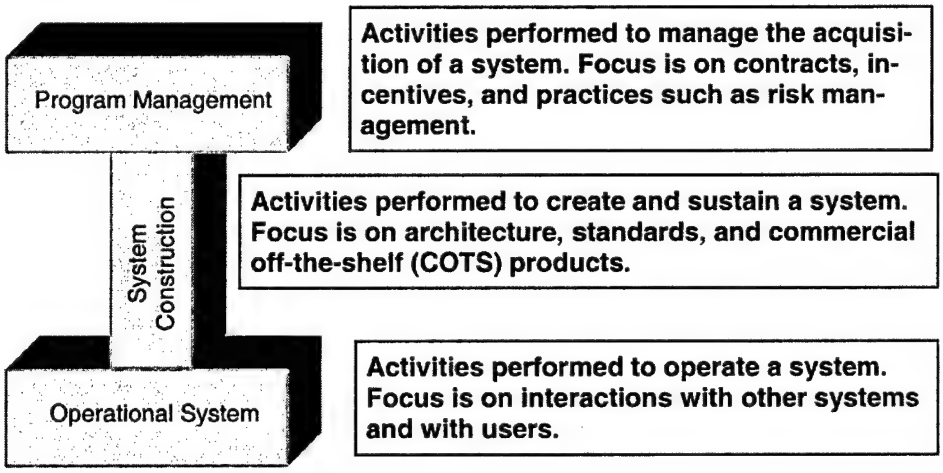


Figure 5: System Activities Model

As shown in Figure 5, *Program Management* defines the activities that manage the acquisition of a system. *System Construction* defines the activities that create or evolve a system (e.g., use of standards and COTS products, architecture). *Operational System* defines the activities within the executing system and between the executing system and its environment, including the interoperation with other systems. The end user is considered part of the operational system.

The simple model represents the activities within a single acquisition program. As illustrated in Figure 6, interoperability between multiple systems requires the development of interoperability between Program Management Offices (PMOs) and the systems they control. This

interoperability will be achieved most effectively through interoperation at the program management, system construction, and operational levels.

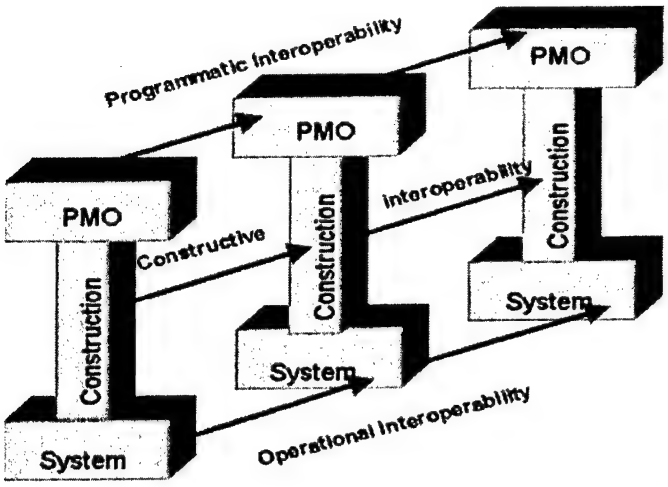


Figure 6: Different Types of Interoperability

7.3.2 Method

The method consisted of three activities: review of the related research, small workshops, and interviews with experts. Each activity is discussed in turn.

The literature survey focused on DoD and related commercial initiatives dedicated to achieving interoperability. Briefings from recent conferences were investigated. A Web-based search was performed to identify related technical literature. Throughout the search process, new leads were identified and pursued. Numerous briefings and papers were reviewed from all sources.

Workshops were held in Washington, D.C. in February and May 2003. The first workshop was held with the SOSI advisory board of DoD experts. A preliminary model of interoperability was presented and feedback on the model was requested in the following areas:

- critical interoperability issues
- insight into programs that are solving critical interoperability problems
- best approaches for conducting research on the current state of the practice

A technical note (CMU/SEI-2003-TN-016) documented the model of interoperability presented and the findings from the workshop.

The second workshop focused on invited presentations representing service perspectives on interoperability. Slide presentations were provided by representatives of the U.S. Army, Navy, Air Force, and National Reconnaissance Office (NRO). Discussion of the different perspectives on interoperability followed.

Finally, a small set of interviews was conducted with experts representing each of the services and NRO. In addition, a representative of the Federal Aviation Administration (FAA) was interviewed. The individuals predominantly represented a technical management perspective. The notes from the interviews were analyzed and coded according to the parameters of the interoperability model. One shortcoming in the interviews pertains to the difficulty we had in identifying end users to provide good feedback from an operational perspective.

7.4 Collaborators

An advisory board of DoD experts was convened for the study. Members include

- Dr. Stan Levine
- Dr. James Linnehan
- Ms. Beth Lynch
- Mr. Chuck Gibson
- Col. Mike Therrien

A number of other experts contributed to this work through their attendance at workshops or by participating in interviews. To ensure confidentiality, these individuals are not identified by name.

7.5 Evaluation Criteria

Evaluation criteria were developed under the initial proposal. Briefly, these can be summarized as

- *The study identifies interoperability problems for which solutions or partial solutions are possible.* Interoperability problems and potential solutions were identified. These are summarized in the Results section of this report. Detailed findings will be published in a technical report.
- *The study validates our vision and model of interoperability, or identifies an alternate model of interoperability supported by lessons learned.* The model of interoperability that has been formulated in this work—namely, the recognition of programmatic, constructive, and operational interoperability—has proven to be of value. The partitioning of elements of a total system, from an acquisition to an operational perspective, has allowed a much larger, and complete, perspective of this work.
- *The study identifies ways in which the SEI can contribute to problems dealing with interoperability.* The lessons learned from this work have significantly influenced future SEI plans. In particular, the COTS-Based System Initiative has grown to become the Integra-

tion of Software-Intensive Systems (ISIS) Initiative. The work on the SOSI IR&D has influenced the program planning for ISIS.

7.6 Results

Despite the large number of organizations involved in addressing interoperability, the problem remains a difficult one. Interoperability problems continue to be significant, even across releases of the same combat system. Any solution will require addressing organizational and technical issues. For example, achieving interoperability between two distinct systems will require changes to management planning and system implementation.

Technical findings according to the programmatic, constructive, and operational dimensions of the model will be published in a final report. The current section addresses general observations on the utility of the model and a preliminary analysis of the interviews.

7.6.1 Observations on the Model

We found that, on the whole, the three-tiered model (programmatic, constructive, operational) was a useful way to organize our investigation and observations. However the model is not complete, because it does not provide a comfortable fit for issues beyond the scope of a program, such as vision, high-level policy, and standards development. Further, some issues fit into more than one tier. For example, different aspects of standards development and application are significant at DoD, programmatic, and constructive levels.

Other perspectives orthogonal to the model were identified (e.g., people oriented, life-cycle oriented). One recommendation was to present the interoperability message from the standpoint of the end users of interoperable systems. A second recommendation centered on the specific activities that must occur in each life-cycle phase to achieve interoperability. In keeping with a people-centered perspective, the life cycle must be extended to include training, fielding, and end users. Our model is silent with respect to life-cycle considerations, but they are implied.

7.6.2 Emergent Themes

The results of the interviews and the workshops are presented according to four major themes.

Definitions

Experts suggest that there are different interpretations of terms such as *system of systems* and *interoperability*, based on divergent needs. What one person considers to be a system of systems, someone else considers a system. This becomes particularly apparent when discussing hugely complex systems, such as the Army Future Combat System, that are really multiple systems of systems.

There is a need for a precise definition of interoperability, because the term can have various meanings in different contexts. For example, interoperability with a weather system may be defined as hourly updates. This could conceivably be handled by a phone call. In contrast, radar reports of objects in the environment need to be updated far more frequently. Thus, interoperability between interacting radar systems may require automated updates as frequently as every few seconds.

The workshop attendees recognized that we may never have precise definitions, partly because our expectations for interoperability are constantly changing. New capabilities and functions offer new opportunities for interactions between systems. This continual change is one of the reasons it is so difficult to define interoperability precisely.[7] Effectively communicating what is meant by interoperability will demand that the context, operations, and requirements be specified whenever the term is used.

Communication

Acquisitions are typically organized around individual systems, but interoperability depends on the quality of communication within and between organizations. Intra- and inter-organizational communication is complicated by a lack of

- understanding who to communicate with
- methodology for how to communicate
- early specification of interoperation ("I built a great little system. I was told to do that piece. I wasn't told about the interface.")
- incentive to pursue interoperability when it makes the work more complex and creates internal and external dependencies that can affect the program

Funding and Control

Experts stressed the contradictions between the objectives for interoperability and current funding models and incentives, which emphasize individual program success for a specific system. They pointed out that interoperability is almost never funded, and reaching agreement between programs is dependent on money: "A key tenet of interoperability is who controls the funds." "Program Executive Offices (PEOs) are reluctant to collaborate because then they will have to share or give up some of their funding." Attendees made the following observations:

- Interoperability (including overhead) must be planned for, funded, and resourced. One industry estimate places the costs to build interoperable systems at 140% of the costs to build similar, non-interoperable systems.

---

[7] The LISI model designates five different levels of sophistication of interation between information systems (C4ISR Architectures Working Group, "Levels of Information Systems Interoperability [LISI]," March 30, 1998).

- Contractors will need to receive incentives to tie a program's success or profit to another program's successes.
- The current funding paradigm will need to change in order to achieve success. "We get [the money] religion real quick. [After all we wouldn't] have done SIAP [Single Integrated Air Picture] without money."
- Program staff is often inexperienced in estimating the costs associated with interoperability. We are not aware of any guidelines for estimating the level of effort necessary to achieve a given level of interoperability.

Beyond money issues, program staff is reluctant to relinquish control for technical reasons. One workshop attendee remarked, "I will be forced to change my perfect implementation... for your imperfect [implementation]." Another added, "Don't take away my control of my stuff."

Leadership Direction and Policy

A barrier to interoperability is a lack of centralized or coordinated ownership of the problem. Short-sighted decisions promote a single system's view at the expense of other systems. Experts also expressed concern about interoperability policy making. Some felt that policies were drafted in a vacuum without a full understanding of the problems and the people affected. They observed that "policy for policy's sake is bad," "policy needs to be sensitive to the implementer's controls and constraints," and "policy needs some flexibility." The following additional comments were made about policy:

- Policy decisions often reflect only a single domain, whereas interoperability may be different in different domains and may require consideration of special constraints (e.g., environment, safety).
- "Writing policy is the easy step. Implementing it is hard." "No one is collecting data to determine which policies are effective." The timing of policy implementation is also critical. "Just because you put out a policy on interoperability does not mean that all the preexisting system or systems under development are going to be interoperable."
- Contractors sometimes prefer standards/policies like DII/COE or the Joint Technical Architecture (JTA) because they are easier to satisfy by "checking a box" instead of having to solve the interoperability problem. Contractors often understand the real interoperability problem, but they prefer not to acknowledge it because then they will have to provide a solution. In this case, policy can work against doing the right thing.

7.6.3 Future State

In addition to considering the current state of practice, we have looked at potential future states. While many programs have some sense of interoperability as one of their goals, it is not clear that the various programs share a vision as to what interoperability really means. While some people appear to have a general goal that amounts to any system being ready at a

moment's notice to interact with another system, the more likely future will involve groups of systems specifically designed to interoperate with each other.

Such collections will be the system of systems for the future. In the design and construction of these collections, there will be a need for the program offices and the contractors (potentially major competitors) to interoperate with each other. While it is reasonable to expect program offices to interoperate (policy can dictate rules that ensure such interoperation), it is less clear how contractors will interoperate at the constructive level.

A key challenge will be to develop an enterprise architectural view. This must define the systems operated by the various enterprise units, the functions of these systems, and the interfaces between them. Such architectures serve as the basis for reasoning about the effects of making new systems operational as well as assisting new programs in understanding the systems with which they must interoperate. It is reasonable to expect that the system architectural vision can be extended to the enterprise (and the contractors) to provide a management architecture. This can then be used to reason about the effects of introducing a new program into the current suite of acquisitions. In both cases, the architecture defines the "as is" enterprise and serves as the basis for measurement as well as understanding what the "to be" should (or could) be.

7.7 Publications and Presentations

There are two publications that describe the work completed under this project, namely workshop presentations from February 2003 and May 2003 and the following technical note:

Levine, Linda; Meyers, B. Craig; Morris, Ed; Place, Patrick; & Plakosh, Daniel. *Proceedings of the System of Systems Interoperability Workshop* (CMU/SEI-2003-TN-016). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tn016.html>.

There were also three workshops held with collaborators in the Washington, DC, area that contributed to the work. Finally, as indicated above, the work of the SOSI IR&D has been reflected in the program plan for the new ISIS initiative.

The material presented here will be expanded, including a discussion of further results and possible future work, in a separate report, *System of Systems Interoperability Independent Research and Development Project: Final Report* (CMU/SEI-2003-TR-024).

Bibliography

All URLs are valid as of September 2003.

[Alexander 64] Alexander, Christopher. *Notes on the Synthesis of Form.* Cambridge, MA: Harvard University Press, 1964.

[Bachmann 00] Bachmann, F.; Bass, L.; Chastek, G.; Donohoe, P.; & Peruzzi, F. *The Architecture Based Design Method* (CMU/SEI-2000-TR-001, ADA375851). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. <http://www.sei.cmu.edu/publications/documents/00.reports/00tr001.html>.

[Bachmann 02] Bachmann, F.; Bass, L.; & Klein, M. *Illuminating the Fundamental Contributors to Software Architecture Quality* (CMU/SEI-2002-TR-025, ADA407778). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2002. <http://www.sei.cmu.edu/publications/documents/02.reports/02tr025.html>.

[Bachmann 03a] Bachmann, F.; Bass, L.; & Klein, M. *Deriving Architectural Tactics: A Step Toward Methodical Architectural Design* (CMU/SEI-2003-TR-004). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. <http://www.sei.cmu.edu/publications/documents/03.reports/03tr004.html>.

[Bachmann 03b] Bachmann, F.; Bass, L.; & Klein, M. *Preliminary Design of ArchE: a Software Architecture Design Assistant* (CMU/SEI-2003-TR-021). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003.

[Bass 03] Bass, Len; Clements, Paul; & Kazman, Rick. *Software Architecture in Practice,* 2nd ed. Boston, MA: Addison Wesley, 2003.

[Brownsword 96] Brownsword, Lisa & Clements, Paul. *A Case Study in Successful Product Line Development* (CMU/SEI-96-TR016). Pittsburgh, PA: Carnegie Mellon University, 1996. <http://www.sei.cmu.edu /publications/documents/96.reports/96.tr.016.html>.

[Clements 02] Clements, Paul; Kazman, Rick; & Klein, Mark. *Evaluating Software Architectures: Methods and Case Studies*. Boston, MA: Addison Wesley, 2002.

[DSPP 00] Director for Strategic Plans and Policy, J5; Strategy Division. *Joint Vision 2020*. Washington, D.C.: U.S. Government Printing Office, 2000. <http://www.dtic.mil/jointvision/jvpub2.htm>.

[Erlikh 00] Erlikh, L. "Leveraging Legacy System Dollars for E-Business." *IEEE IT Professional* 2, 3 (May/June 2000): 17-23.

[Friedman-Hill 03] Friedman-Hill, Ernest. *Jess in Action*. Greenwich, CT: Manning Publications Company, 2003.

[Lehman 80] Lehman. M. M. "On Understanding Laws, Evolution and Conservation in the Large Program Life Cycle." *Journal of Systems and Software 1*, 3 (1980): 213-221.

[Lehman 85] Lehman, M. M. & Belady, L. *Program Evolution: Processes of Software Change*. London: Academic Press, 1985.

[Müller 94] Müller, H. A.; Wong, K.; & Tilley, S. R. "Understanding Software Systems Using Reverse Engineering Technology," 41-48. *The 62nd Congress of L'Association Canadienne Francaise pour l'Avancement des Sciences Proceedings* (ACFAS 1994). Montreal, Quebec, May 16-17, 1994.

[Perry 98] Perry, Dewayne. "Generic Architecture Descriptions for Product Lines." *ARES II: Software Architectures for Product Families 1998.* Los Palmos, Gran Canaria, Spain, February 1998. <http://www.ece.utexas.edu/~perry/work/papers/ares2.pdf>.

[Sommerville 00] Sommerville, I. *Software Engineering*, 6th ed. Harlow, England: Addison-Wesley, 2000.

[Tekinerdogan 00] Tekinerdogan, Bedir. "Synthesis-Based Software Architecture Design." PhD Thesis, University of Twente, March, 2000.

[Ulrich 90] Ulrich, W. M. "The Evolutionary Growth of Software Engineering and the Decade Ahead." *American Programmer 3*, 10 (1990): 12-20.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	September 2003	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
SEI Independent Research and Development Projects	F19628-00-C-0003

6. AUTHOR(S)

Felix Bachmann, Len Bass, David Carney, Sven Dietrich, Peter Feiler, Suzanne Garcia, Mark Klein, Tony Lattanze, John McHugh, B. Craig Meyers, Ed Morris, Patrick R. H. Place, Dan Plakosh, Robert Seacord

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2003-TR-019

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	ESC-TR-2003-019

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

Each year, the Software Engineering Institute (SEI) undertakes several Independent Research and Development (IR&D) projects. These projects serve to (*a*) support feasibility studies investigating whether further work by the SEI would be of potential benefit and (*b*)support further exploratory work to determine if there is sufficient value in eventually funding the feasibility study work as an SEI initiative. Projects are chosen based on their potential to mature and/or transition software engineering practices, develop information that will help in deciding whether further work is worth funding, and set new directions for SEI work. This report describes the IR&D projects that were conducted during fiscal year 2003 (October 2002 through September 2003).

14. SUBJECT TERMS	15. NUMBER OF PAGES
acquisition simulation, role playing, acquisition training, expert systems, architecture design, architecture analysis, model-based reference architectures, wireless device security, sustainment, sustaining legacy systems, sustainability, interoperability, system of systems	68

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL